Categories, Mathematics, and Systems

Larry A. Lambe[1], PhD, PhD (Hon)
MSSRC[2]
llambe@mssrc.com
http://www.mssrc.com

March, 2018

[1]Chief Scientist, Multidisciplinary Software Systems Research Corporation

[2]The speaker is grateful to Professor Ronnie Brown, Bangor University, Wales, UK for many years of useful conversations about mathematics

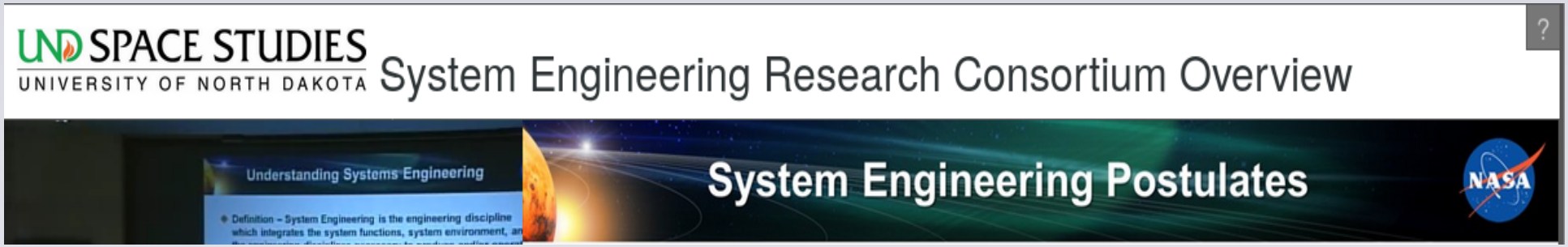
Introduction

- A previous presentation with the same title as this one was given in January of this year for the January 2018 - Workshop at IW18,
- You are encouraged to see the penultimate pdf file toward the bottom of the page and many other interesting pdf files at **https://sites.google.com/site/sswg2018/iw**.
- This presentation will not be exactly the same, but in terms of information, it will contain and extend what was covered in the talk above.
- The goal is to provide a background for understanding language that is used in a growing number of works dealing with ideas about systems in terms of what is called **category theory**.
- A knowledge of category theory is not assumed here. Basic ideas from that subject will be presented in an intuitive way.

Some History

- Rosen, R. (1958), The representation of biological systems from the standpoint of the Theory of Categories, Bull. Math. Biophys. 20, 245-260.
- Bertalanffy, L. von (1973), General System Theory, Harmondsworth, Penguin.
- Louie, A.H. (1983), Categorical System Theory, Bull. Math. Biol. 45, 1029-1072.
- Salthe, S.N. (1985), Evolving hierarchical systems: their structure and representation, Columbia University Press.
- Rosen, R. (1986), Theoretical Biology and complexity, Academic Press.
- Andrée Ehresmann and Jean-Paul Vanbremeersch (2007), Memory Evolutive Systems, Elsivier, Studies in Multidisciplinarity Volume 4.
- Breiner, Spencer et al. (2018), Disciplinary Convergence in Systems Engineering Research, Springer, Cham, 449–463,

Michael D Watson: http://space.edu/colloquium.aspx



- **Postulate 2: The Systems Engineering domain consists of subsystems, their interactions among themselves, and their interactions with the system environment**
- **Postulate 7: Understanding of the system evolves as the system development or operation progresses**

Objectives

- Present some preliminary mathematical notions (set theory, directed graphs, algebra)
- Define **category** in the mathematical sense
- Give examples of some systems
- Present category theory basics with more examples
- Give a **working definition** of "system" in terms of the language of category theory using intuition and Postulate 2
- Begin to incorporate other postulates into the working definition – specifically Postulate 7
- **Questions and Answers**

Some Set Theory

- A Set is a collection of things called *elements*
- – the order in which they are listed is irrelevant.
- $\mathbf{X} = \{\mathbf{Q}, \mathbf{1}, \mathbf{a}, \mathbf{2}, \mathbf{z}, \mathbf{t}, \mathbf{21}, \mathbf{Y}\}$ is a set of elements that are numbers and letters.
- **A function**

$$\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$$

from one set to another is a correspondence between the elements of $\mathbf{X}$ and those of $\mathbf{Y}$

- **such that to each elements $\mathbf{x} \in \mathbf{X}$ there corresponds one and only one element $\mathbf{f}(\mathbf{x}) \in \mathbf{Y}$.**

Some Set Theory (Contd)

- Consider

$$f : \{Q, 1, a, 2, z, t, 21, Y\} \to \{\text{number}, \text{letter}\}$$

Definition:

$$f^{-1}(y) = \{x \,|\, f(x) = y\} \text{ so}$$

- $f^{-1}(\text{number}) = \{1, 2, 21\}$
- $f^{-1}(\text{letter}) = \{Q, a, z, t, Y\}$
- So X is partitioned by

$$X = f^{-1}(\text{number}) \amalg f^{-1}(\text{letter}).$$

Classification

- So functions can be used to classify the elements of a set.
- The fact that a function is required to be **single-valued** is interesting in this context.
- Suppose that you have a set of elements that are red, green, round, square, etc.
- What if an element if both red and round?
- One solution: Take

$$Y = \{\text{red}, \text{green}, \text{round}, (\text{red}, \text{round}), (\text{red}, \text{square}), ...\}$$

- By satisfying the definition this way, we have created *hierarchies*.
- This is not the only way to solve the problem.

Relations and Functions

- So any function f partitions X into disjoint subsets and thus a function defines a decomposition of its source X.

$$X = \coprod_{y \in Y} f^{-1}(y).$$

- To review, the decomposition by inverse images is actually a partition: if there is a z in both $f^{-1}(y)$ and $f^{-1}(y')$, then $f(z) = y$ and $f(z) = y'$ so **by the definition of function**, $y = y'$.

- This is why functions $O \to T$ classify the elements in O.

Equivalence Relations and Functions

- A relation r on X is any association of one element in X to another element. We can denote such a comparison by

$$x \xrightarrow[r]{} x'$$

- A function f with source X defines a relation that satisfies
- $\mathbf{x} \xrightarrow[\mathbf{f}]{} \mathbf{x}'$ if and only if $\mathbf{f(x)=f(x')}$
- Clearly $x \xrightarrow[f]{} x$ (reflexive),
- if $x \xrightarrow[f]{} x'$ then $x' \xrightarrow[f]{} x$ (symmetric), and
- if $x \xrightarrow[f]{} x'$ and $x' \xrightarrow[f]{} x''$ then $x \xrightarrow[f]{} x''$ (transitive).
- We call a general relation r that satisfies all three of the above conditions even without the aid of a function, an *equivalence relation*.

Equivalence Relations and Functions (Cont'd)

- Just as a function gives a partition of X, an equivalence relation also gives a partition on X.
- We define an "equivalence class" for $x \in X$ to be $[x] = \left\{ x' \in X \mid x \xrightarrow[r]{} x' \right\}$.
- In analogy with functions, we have that the sets $[x]$ for $x \in X$ form a partition of X
- and we have a function

$$X \xrightarrow{q} X/r$$

where $X/r = \{[x] \mid x \in X\}$ (*quotient set*).

- **This important construction gives us a type of glue to create new spaces from old.**

MSSRC

Monoids

Figure: One node, label 0; one arrow, label 1

- There is an operation on the graph.
- Starting at the node 0 at the bottom, and following the loop in the direction of the arrow all the way around gives one iteration of the operation.
- We denote the operation by the symbol "$+$".
- So one iteration gives "1+1".

Monoids (Contd)

Figure: One node, label 0; one arrow, label 1

- You know this monoid from childhood. It is the non-negative numbers

$$\mathbb{N} = \{0, 1, 2, \ldots, \}$$

  with the ordinary operation of addition.
- The node 0 acts as an "identity element": $0 + n = n + 0 = n$.
- The directed graph with one node 0 and one arrow "1'is a compact representation of this monoid.

MSSRC

Monoids (Contd)

Figure: One node, label 1; one arrow, label a

- We can play the same game with this loop, but now, we can denote the operation of going around the loop by juxtaposition and the node at the bottom by 1.
- So going around n-times will give $aaa \ldots aa$ and we will denote that by a^n.

Monoids (Contd)

Figure: One node, label 1; one arrow, label a

- So we have a monoid of the form

$$\mathbb{M} = \{1, a, a^2, a^3, \dots\}$$

with multiplication given by juxtaposition and identity element 1.

- Note that path iteration gives $a^n a^m = a^{n+m}$.

Monoid and Words

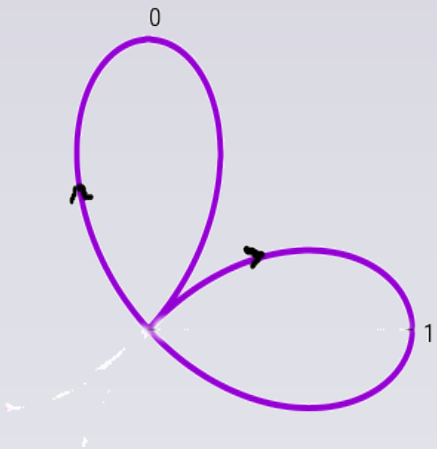


Figure: One node and two arrows (compose multiplicatively).

- This time, we have all "words" in 0 and 1.
- That means that this little digraph essentially holds all recorded history that has ever been written or will be.
- $0100100001100101011011000110110001101111$
- $011101110110011101110010011011000110010 0$
- (The above reads: 'Hello world')

MSSRC

A New Machine

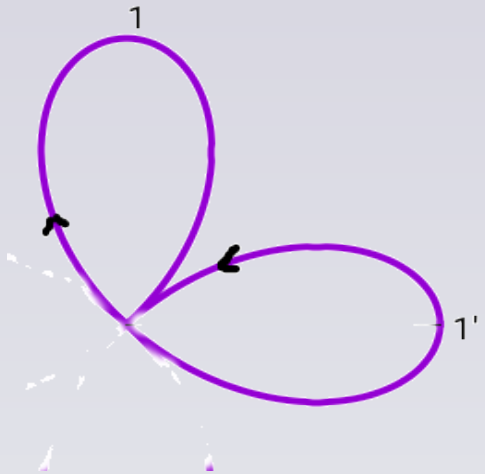


Figure: One node and two arrows one reversed.

- This time, we have a reverse path available so we have words like $n + n'$ and $n' + n$.
- What if we wanted n' be the additive inverse of n? so that the above paths would be zero?
- We would need to "add the equations"

$$\begin{aligned} n + n' &= n' + n \\ n + n' &= 0. \end{aligned}$$

MSSRC

A New Machine (Contd)

- Can we just "add equations" like that?
- It might be better to **define relations**

$$(n + n') \xrightarrow[r]{} 0 \ , \qquad (n' + n) \xrightarrow[r]{} 0$$

- and take the quotient using the construction from an earlier slide – *if they lead to* an equivalence relation.
- This brings up an issue that may be important for applications to engineering.
- This problem is obviously phrased in terms of mathematics but the process is much more general.
- So let me explain what I think that engineers who may be interested in learning category theory should know.

A New Machine (Discussion)

- Let us suppose that we have a fixed set of rules for constructing some type of system. We get together and intuit some specs. We'd like to check that our specs satisfy all the necessary rules given in the fixed case.
- If our fixed rules are clear, there ought to be a way we can go down a list checking that everything we need to be true is true. If not,
- then we could to derive the necessary checks using just our intuitive list of specs. If that fails,
- the next best thing would be a precise algorithm (mathematically sound procedure) for adding new things to the original list to be consistent with it and at the same time, fulfill the requirements of the fixed list.
- If that cannot be done, then our original list needs to be reconsidered.

A New Machine (Discussion Contd)

- My example is such that our fixed set of rules is that of a quotient set X/r where r is an equivalence relation.
- The fixed rules are
  - reflexive
  - symmetric
  - transitive
- We have seen that there is a universal function

$$X \xrightarrow{q} X/r \ .$$

- In the previous example, it classifies elements of X according to "traits" specified in he set X/r.
- Here we have a different application, but the **process** is the same.

MSSRC

A New Machine (Discussion Contd)

- The way this application works is that we start with the intuition that if we add a reverse path in the one loop monoid, it should act as the inverse operation.
- That comes from the thought that if we go around the loop twice carrying a rope, when we get back to the node, we cannot pull the rope in without letting go,
- but if we go around the "up" path and then the "down path", when we get back home, we can pull the rope back without letting go.
- Since we are using the "additive notation" for this monoid, we see this as saying the composite path $1 + 1'$ is the zero path 0
- i.e., the path that goes nowhere or the identity path.

A New Machine (Discussion Contd)

- In our two loop situation, that intuition is not strictly true, but if we had a one loop model with the loop reversible (two way traffic), it would be true.
- I just did not want to make a new notation for that (digraphs are what they are).
- So we construct a big object that has all (non-commutative) words in the symbols 1 and $1'$ like
  $1 + 1' + 1 + 1' + 1 + 1' + 1 + 1' + 1 + 1' + 1'$
- we can't gather the 1s together and/or the $1'$s together because we cannot commute a 1 past a $1'$.
- However, if we start with the intuition that $1 + 1'$ should be 0 *and* $1 + 1'$ should be equivalent to $1' + 1$

MSSRC

A New Machine (Discussion Contd)

- i.e. make it an equivalence, then if we can extend this notion to the full set of requirements i.e., if the intuitive rule extends to an equivalence relation r *uniquely*,
- then we can form W/r and inside W/r, we can commute 1s past $1'$s
- and so everything "straightens out" to n 1s (which we call n) or k $1'$s which we call $-k$
- and so we have negative numbers added onto the non-negative ones using this construction.
- I don't know if that is how integers come about in nature, I only know that this is one way to make them.

One More Digraph

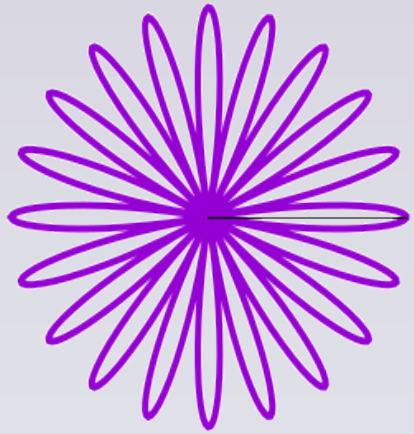


Figure: One node, ten arrows all in one direction (multiplicatively).

- This time, we have all "words" in $a, b, \ldots, j$. and we add the relations $ab \xrightarrow[com]{} ba$ and so on for all the "generators" $a, b, \ldots, j$.
- Since we now can gather like elements together, we have all words of the form $a^{i_1} b^{i_2} c^{i_3} \ldots j^{i_{10}}$
- Composition now is juxtaposition followed by perhaps many commutations (if you like bookkeeping).

MSSRC

General Monoids

- The monoids we have looked at without consideration of relations among the associated set of words are *free* monoids – called so because they are "free" of any extra relations.
- Our monoid above has the relation that all elements commute, i.e. $xy = yx$ for all paths x and y. For that reason, it is called the free commutative monoid (on as many "generators" as there are loops).
- All monoids are quotients by a equivalence relation on a free monoid with a given number of loops
- where the equivalence relation is generated by a set of "word equivalences".
- There is a vast mathematical theory and ongoing research in the area of "rewrite rule theory" associated to the analysis of the mathematical structure of such quotient monoids.

General Monoids and Groups

- The integer models (two loops with opposite arrows) gave us the integers.
- Generally, if a monoid has inverses, it is called a group.
- Groups occur all over the place in nature and in art as **symmetry groups**.
- One can think of all rotations and or translations of space which leave an object fixed.
- That collection of transformations form a symmetry group of the object.
- For example, there are rotations of the plane that map a square into itself.
- Symmetry groups are important in crystallography and helps us to understand molecular structure among other things.

General Monoids and Groups (Contd)

- Just like monoids, groups may be free or have relations.
- All groups are given as a quotient set of a free group by relations.
- Again, the word problem is an issue. So a bit more about it would be in order here because it will resurface when we look at categories.
- One can view relations in a free monoid (group) as an arrow as we did before with the two loop monoid.
- For example, we might have $w_1 = w_2$ as a relation (before it was $ab = ba$), etc.
- We can go through a list of "free words" and exchange every word that has w_1 by w_2.
- We can do that with every "equation" in our list of relations.

General Monoids and Groups (Contd)

- Now as we go through the list over and over again, there are two questions that stand out:
  1. Does the rewriting ever stop?
  2. If it stops, could it stop in two different words at different times?
- That is the **word problem** an as mentioned, it is "well-studied".
- If the word problem has a solution, the unique word at the end of all the rewriting is called a **normal form** of the original word.
- There are various programs (including those in the references about to be cited) that can help with rewriting and a list is given next.
- The list is by no means complete, but it is a good start for anyone interested in delving deeper into the subject.

MSSRC

Entry Into Rewriting Theory and Normal Forms

Note: All programs listed below are very technical and do not have the type of user interface that might be expected of typical application programs. They are mostly research tools for experts working in the various areas of expertise they address

- **http://mssrc.com/lambe/** (Gareth Evans' PhD thesis at the lower right hand column)
- **http://servus.math.su.se/bergman/** (Jörgen Backelin on Noncommutative Gröbner Bases)
- **https://www.gap-system.org/** (Generators, relations, and normal forms and more)
- **https://faculty.math.illinois.edu/Macaulay2/** (Gröbner Bases and Algebraic Geometry)
- **https://arxiv.org/abs/math/9903032** (Anne Heyworth and Ronnie Brown on Left Kan Extensions)
- **https://fricas.github.io/** (FriCAS, Dick Jenks et al, free monoids, Gröbner bases, and much more in terms of abstract mathematics in general)

Categories

- We can give a definition at this point which should make sense in terms of what has been presented so far.
- **A category $\mathbf{C}$ is a directed graph whose nodes are one loop monoids**. (Note that it is assumed that all composite paths are considered as being in the category as we did for loops)[3]
- It is not likely that many sources in the (mathematics) literature will define them this way, but they will be equivalent to this one with possibly some necessary language adjustment.
- From our point of view, you may think of a monoid as a one node (object) category or a category as a many node (object) monoid.

[3] Relations may be allowed among the arrows.

Categories (Contd)

- **Standard Notation**: For a category C

  node $=$ object; arrow $=$ morphism.
- The collection of nodes (objects) is $\mathrm{Ob}(C)$
- The collection of arrows (morphisms) is $\mathrm{Arr}(C)$
- The set of all arrows from a node a to a node b in C is denoted by
$$\hom_C(a, b).$$
- Thus, the set $\hom_C(a, a)$ is a monoid.
- The identity element is denoted by 1_a.
- When the context is clear, we often simply write $\hom_C(a, b)$ as $[a, b]$ or $[a, b]_C$ for more specificity.

Example I: An I/O Chain System

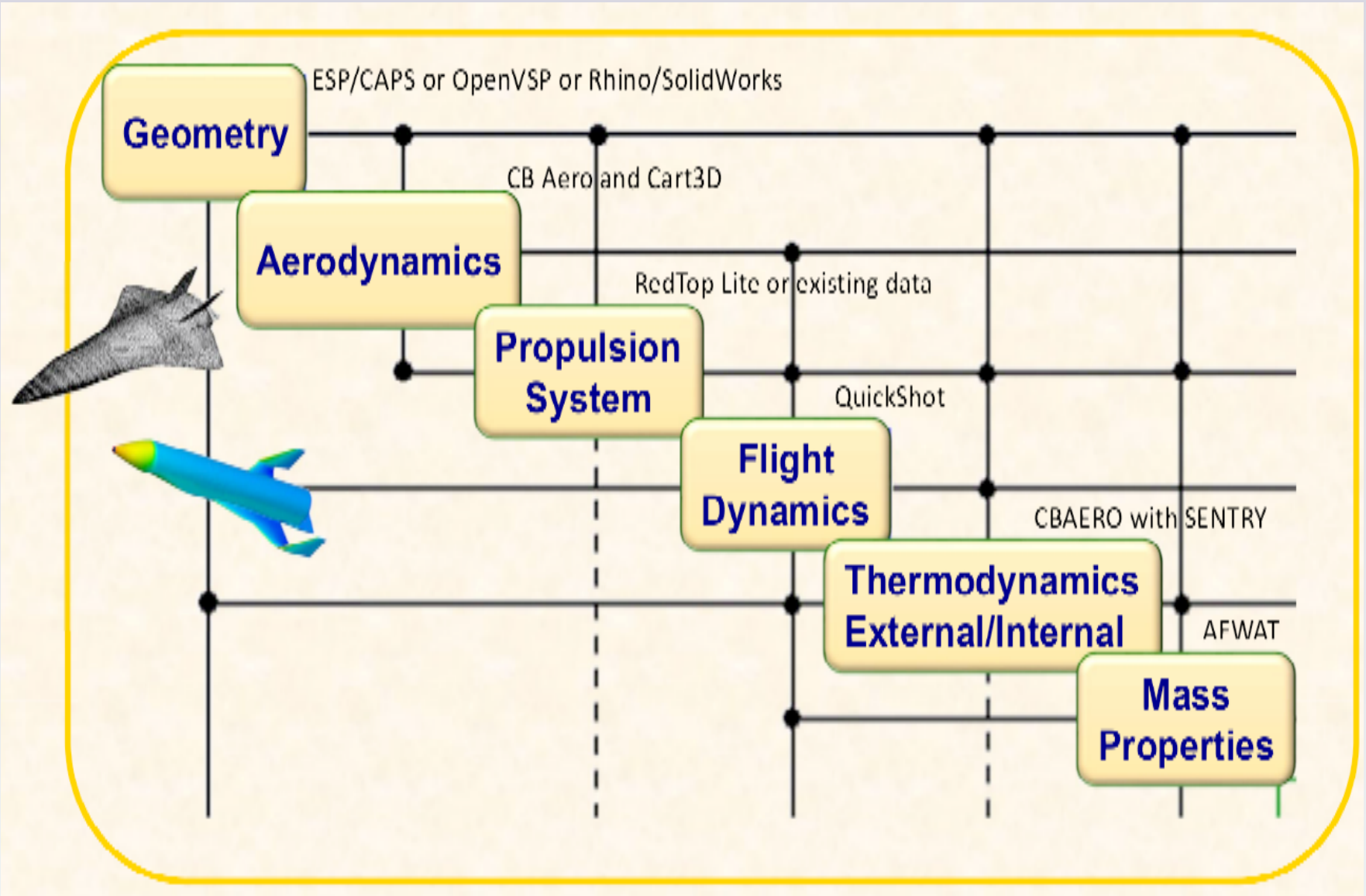


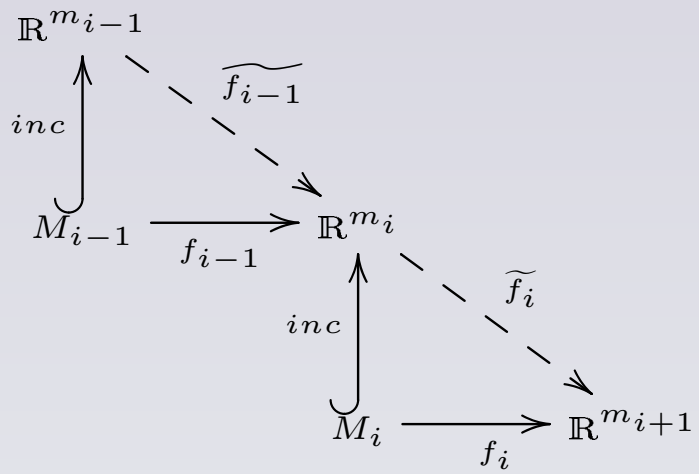
Figure: From a Phase III SBIR on Hypersonic Systems MDA

An I/O Chain (Contd)

- A simple digraph describes the situation, viz.

- The nodes represent triangulation of spaces upon which the indicated solvers are defined.
- However, the arrows are *not* functions between the nodes because the solver at node i may have output that does not fall on the mesh defined at node $i + 1$.
- So what are the arrows?

An I/O Chain (Contd) [4]



- The node $\bullet_i$ is the mesh M_i and the arrow is the triangular system you see above for M_i.
- The f_i are thought of as "training sets" and the $\widetilde{f_i}$ are Gaußian process regression models based on the f_i.
- This is a typical MDA situation, but there are many other scenarios.

[4] I am grateful to Dr. Richard Snyder, AFRL/WPAFB for conversations concerning these ideas (2009)

Some Preparation

- Everything we have discussed so far has been within the category of Sets except for I/O chains where the arrows were not function.
- A definition will be given next which may be represented in many different categories, but the actual **construction** of the object described
- may be done using "glue" like the one we have spent so much time working with so far (in set theory).
- The set theory examples give us intuition for what we may guess in other categories which may be as far away from set theory as we can imagine.
- This makes category theory even more powerful as we gain more and more intuition by working in different categories as the need arises.

A Definition in English

- In almost every paper I have read with the word "systems" and the phrase "category theory" occurring, there is also present the word
- *colimit*.
- So one of the main points regarding category theory I want to get across today is that
- **a colimit is intuitively an object that has been obtained by "machining together" a collection of parts using a collection of "compatibility/consistency" instructions.**

Example II: A Diagram System

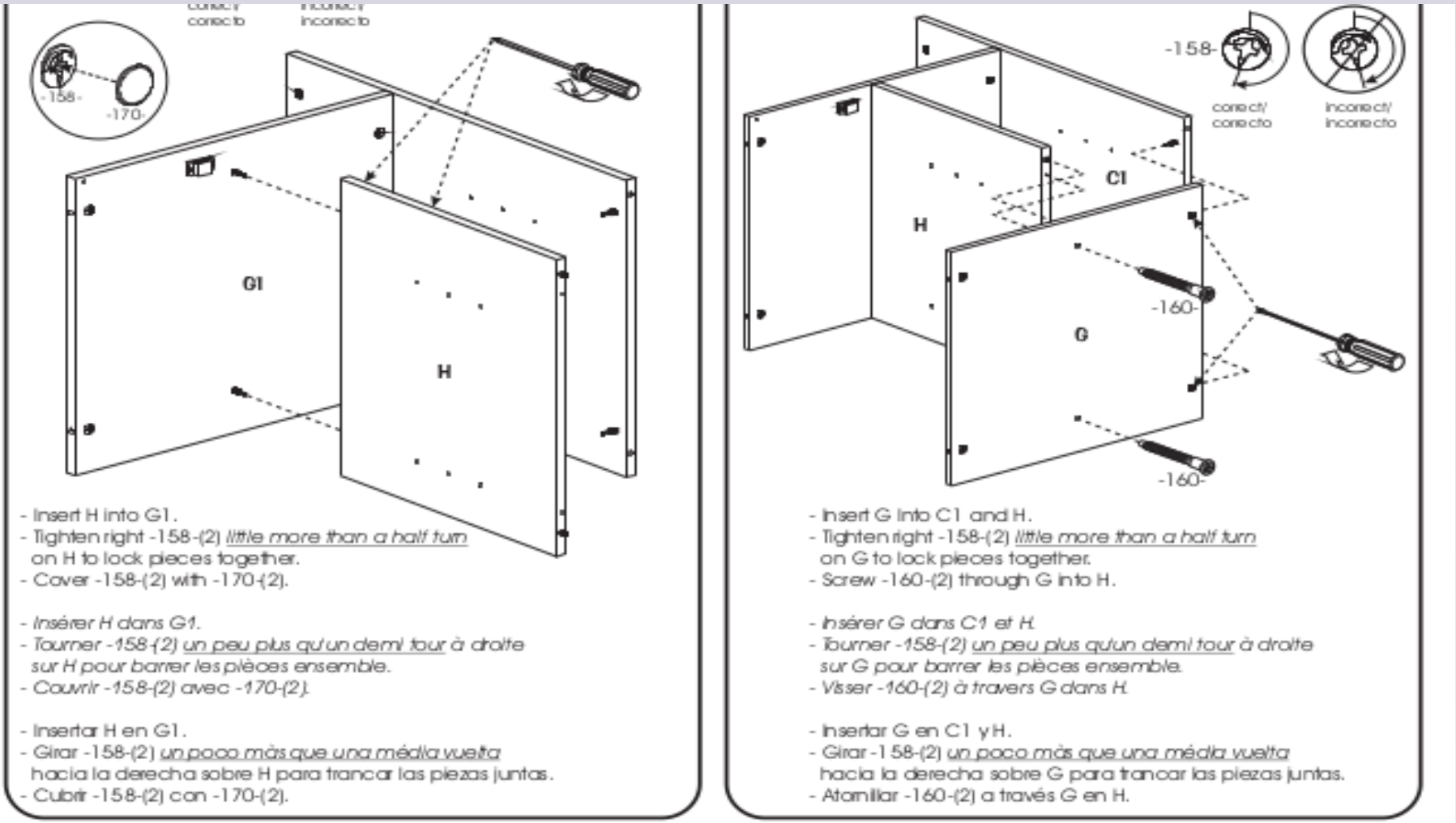


Figure: From my printer stand instructions

MSSRC

Example III: A Human System

Figure: Leonardo da Vinci (1492): Gallerie dell'Accademia, Veni

Example IV: Equivalence Relations

- Recall that an equivalence relation on a set X is a relation which is determined by a distinguished subset of arrows of the form $x \xrightarrow[r_{x,x'}]{} x'$
- For such an arrow, let $src(r_{x,x'}) = x$ and $tgt(r_{x,x'}) = x'$.
- Let N be the set of all source nodes and all target nodes
- and let the set of all arrows be the collection of all $r_{x,x'}$ for $x, x' \in N$ be denoted by $arr(N)$.
- Then $\underline{N} = (N, arr(N))$ is a directed graph.
- The equivalence relation axioms make $\underline{N}$ into a category!
- The hom sets $hom_{\underline{N}}(x, x)$ are monoids because $x \xrightarrow[r_{x,x}]{} x$ acts as the identity element and
- transitivity says that products are associative.

MSSRC

Example IV: Equivalence Relations (Contd)

- But there is something extra here: symmetry says that every arrow (morphism or path – depending on what language you like) is reversible.
- In other words, every arrow is invertible that is, has an inverse.
- This is a special kind of category called a **groupoid**.
- So a groupoid is a category in which every arrow is invertible.
- **Monoids are to groups as categories are to groupoids**.
- So from our point of view, you may think of a group as a one node groupoid or a groupoid as a many node group.
- There is one more important category to consider.

Isomorphism (Back to Our Loops)

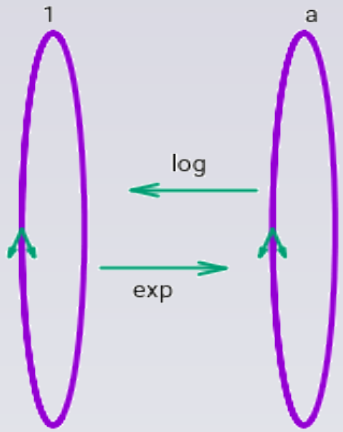


Figure: Left, node = 0, arrow = 1, right, node = 1, arrow = a

- We've seen that the monoid at the left has path products
- $0, 1, 2, \cdots, n \ldots$ and an operation denoted by $+$.
- The one on the right has path products $1, a, a^2, \ldots, a^n, \ldots$.

Isomorphism (Contd)

- These monoids are, in fact isomorphic (they are one-to-one and preserve path products). The isomorphisms are given by
- on nodes:

$$\begin{aligned} \log_a : 1 &\mapsto 0 \\ \exp_a : 0 &\mapsto 1 \end{aligned}$$

- on morphisms

$$\begin{aligned} \log_a(a^n) &= n \\ \exp_a(n) &= a^n. \end{aligned}$$

  notice that we needed two kinds of correspondences for this isomorphism.
- One on objects,
- one on morphisms.

MSSRC

Functors

A functor from one category $\underline{C}$ to another $\underline{D}$ is a functional correspondence between nodes and arrows of one with the nodes and arrows of the other.

- A functor $F : \underline{C} \rightarrow \underline{D}$ consists of
- $$\mathbf{F : Obj(\underline{C}) \rightarrow Obj(\underline{D})}$$

  and for all objects $a, b \in Obj(\underline{C})$, a correspondence

  $$\mathbf{F : hom_{\underline{C}}(a, b) \rightarrow hom_{\underline{D}}(F(a), F(b))}$$

- which preserves composition:
- $F(gf) = F(g)F(f)$

$$F\left((a \rightarrow b) \circ (b \rightarrow c)\right) = F\left((a \rightarrow b)\right) \circ F\left((b \rightarrow c)\right).$$

MSSRC

Example V: The Category of Sets

- The nodes of $\underline{\text{Set}}$ are all sets.
- The arrows (hom sets) $hom_{\underline{\text{Set}}}(X, Y)$ are the sets of all functions from X to Y.
- Clearly, composition of functions is associative

$$(fg)(h(x)) = f(g(h(x))) = f(gh(x))$$

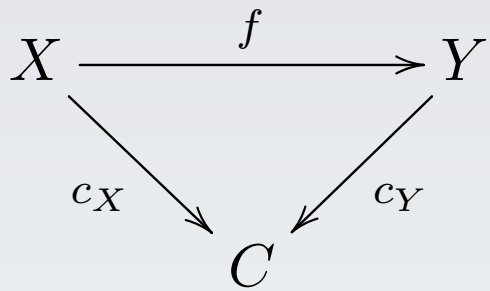
  for all x
- so $fg(h) = f(gh)$, and
- the identity function 1_X is the identity for composition of functions.

Example V: The Category of Sets (Contd)

- Thus, $\underline{\mathsf{Set}}$ is indeed a digraph in which every node defines a monoid $hom_{\underline{\mathsf{Set}}}(X, X)$.
- The category of sets plays an important role in category theory.
- Thus, apparently, categories are more familiar than one thinks at first —
- OR perhaps, sets are more abstract and interesting than one thinks at first.
- **One power of category theory is that we can ignore what is "inside" a node (object) and see what can be done with only a graph structure having a monoid at every node and a generalized partial product structure globally.**

Cocones

- A **pre-cocone** in a category $\underline{C}$ is a functor $\underline{I} \xrightarrow{F} \underline{C}$.
- Sometimes a pre-cocone is just called a "diagram".
- A **cocone over F** is a collection of morphisms from a pre-cocone to a node C such that "all relevant morphisms commute", i.e.
- for every node Z in the image of F (a diagram in $\underline{C}$) there is a morphism $Z \xrightarrow{c_Z} C$ such that



- commutes (remember, f is in the image of F).

Cocones (Cont)

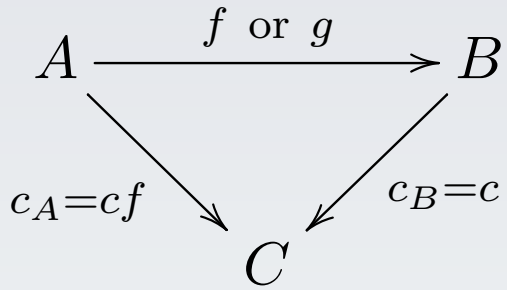
- As a little exercise, it is easy to see that a cocone over a functor from the category

$$\bullet \rightrightarrows \bullet \,,$$

- is **equivalent to** a diagram

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{c} C$$

- with $cf = cg$ because that is equivalent to the cocone
-



Example Of a Colimit

- Consider the situation

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{c} C$$

- which we may consider to be a cocone from the previous frames if

$$cf = cg \text{ holds.}$$

- It is called a **final cocone** if it is unique in the following sense: If given another cocone (so $df = dg$)

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{d} X$$

- then there is a unique arrow $C \xrightarrow{\varphi} X$ (with $d = \varphi c$) – and then C is called a *coequalizer* which in this case is a **colimit**.

MSSRC

Example Of a Colimit (Contd)

- So how does it work? We want C such that

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{c} C \text{ such that } cf(a) = cg(a)$$

  with the universal property.
- So why not "make" $f(x) = g(x)$ in B?
- That is, define a relation $f(a) \xrightarrow[R]{} g(a)$ in B.
- and take $C = B/R$, the quotient set.
- The trouble is that it might not be an equivalence relation.
- We have been through this in English in an earlier frame:
- we need to discuss how to make it one.
- So how could can this be done?
- Let's state the problem clearly.

MSSRC

Example Of a Colimit (Contd)

- We have a situation

$$A \underset{g}{\overset{f}{\rightrightarrows}} B$$

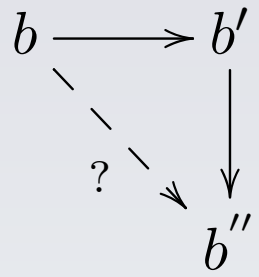
- and we define a relation

$$R = \{(b, b') \in B \times B \mid \text{ for some } a \in A, b = f(a), b' = g(a)\}.$$

- and check to see if it is an equivalence relation. So
- R reflexive? Yes, if we add the diagonal $\Delta = \{(b, b)\}$.
- R symmetric? Yes, if we add $R^{op} = \{(b', b) \mid (b, b') \in R\}$.
- So far, we have $R_1 = R \cup \Delta \cup R^{op}$,
- Is it now transitive? Hmmmm.
- Let's see what that means.

Example Of a Colimit (Contd)

- So far, we have R_1, the reflexive and symmetric "closure" of R.
- We want to see if $b \longrightarrow b'$ and $b' \longrightarrow b''$ then $b \longrightarrow b''$ (all in R_1).
- Let's go to the graph:



- Well, if the dotted arrow isn't there, add it to R_1.
- We get an equivalence relation, but at what cost?
- Answer: It depends on what category you are in.

Example Of a Colimit (Contd)

- The type of colimit above is called an **coequalizer**.
- Here is a special case:

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{c} C$$

- where both f and g are inclusion maps.
- The colimit in this case, "glues" the image of f to the image of g in B to get $C = B/(f(a) = g(a))$.
- The result (**and the work to get it**) depends on the category you are in.
- I will do an example in $\underline{HoTop}$, the category of topological spaces in light of an equivalence relation called homotopy.
- For this, A is a figure eight (one point union of two circles),
- and $B = S^1 \times S^1$, the torus.
- f is the inclusion of the figure eight (picture follows).
- g is the inclusion of the point of intersection of the two circles.

MSSRC

A Topological Colimit

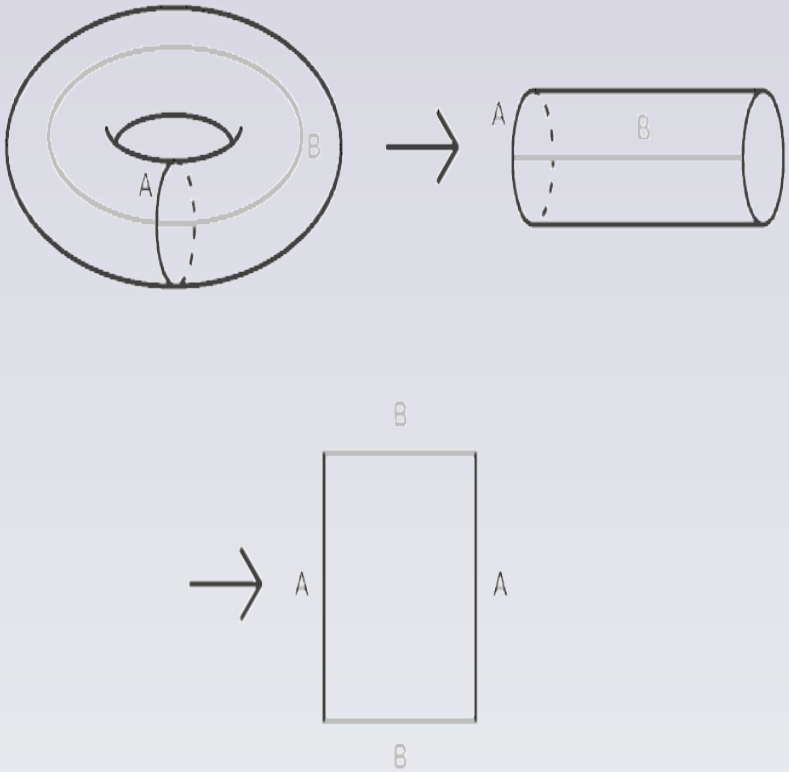


Figure: Step 1: cut apart the torus, but keep track of things

- step 2: deform the rectangle into a disk continuously.
- The figure eight has become the boundary of the rectangle and hence the disk.

MSSRC

A Topological Colimit $S^2 = S^1 \times S^1 / S^1 \wedge S^1$

- Step 3: Shrink the boundary of the disk to a point,
- but you can only stretch the interior of the disk and not contract the whole thing to a point –just the boundary circle.

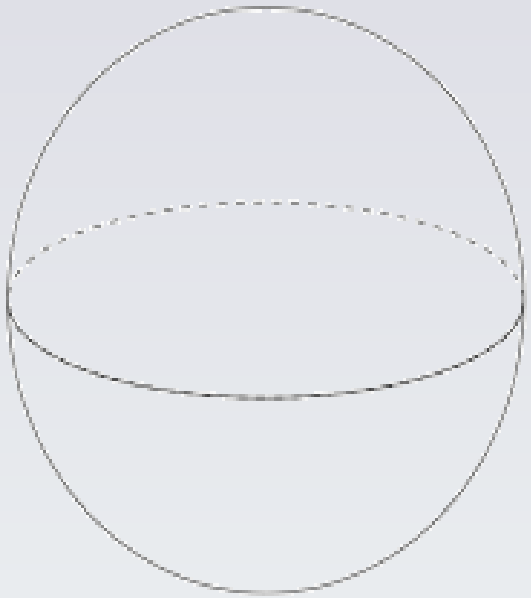


Figure: The result of Step 3: A 2-Sphere

A Working Definition of System

A system is either a category or a final cocone (vertex) whose nodes are recursively either an atomic category or a system.

Comments:

- This definition was inspired by Watson's postulates 2 and 7 only. As such, it is probably too general.
- Postulate 7 might be covered by allowing nodes to be petri-nets considered as (enhanced) categories.
- Obviously, the working definition depends of the category in which it is to be applied.
- It is hoped that this definition will evolve in interesting ways after community discussions.

The Hom Functor (Covariant)

$$\underline{C} \xrightarrow{h^A} \underline{\mathrm{Set}}$$

$$X \xmapsto{h^A} [A, X]_{\underline{C}}$$

$$\begin{array}{ccc} B & & [A,B]_{\underline{C}} \\ \downarrow{\scriptstyle f} & \xmapsto{h^A} & \downarrow{\scriptstyle h^A(f)} \\ B' & & [A,B']_{\underline{C}} \end{array}$$

Scratch Pad

$$\begin{array}{ccc} A & \xrightarrow{\text{given } \xi} & B \\ & \searrow^{f\xi} & \downarrow{\scriptstyle f} \\ & & B' \end{array}$$

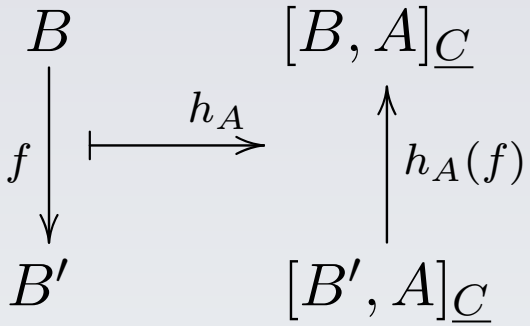
Thus, we take $\mathbf{h^A(f)(\xi) = f\xi}$.

MSSRC

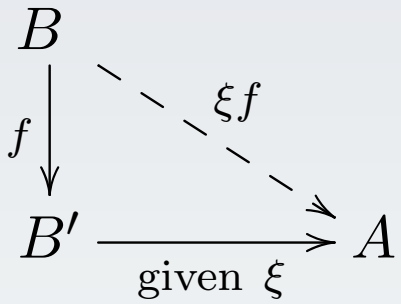
The Hom Functor (Contravariant)

$$\underline{C} \xrightarrow{h_A} \underline{\mathrm{Set}}$$

$$X \xmapsto{h_A} [X, A]$$



Scratch Pad



Thus, we take $\mathbf{h_A(f)(\xi) = \xi f}$.

The Opposite Category

- We've seen an example of a functor that reverses composition. Such functors called contravariant functors, but there is no need for another name!
- Given a category $\underline{C}$, define the **opposite** category $\underline{C}^{op}$
- to have the same nodes as $\underline{C}$, but whose arrows are the arrows of $\underline{C}$ reversed.
- A contravariant functor $\underline{C} \to \underline{D}$ is the same as a (covariant) functor $\underline{C^{op}} \to \underline{D}$.

> *There once was a boy from the farm*
> *who thought that modules had charm*
> *if modules are cute, then who could dispute*
> *by reversing my arrows – no harm!*

- (From L. Lambe, U of I, Chicago: A limerick to an office mate from Goshen, IN who was writing his PhD thesis on coalgebras and comodules in graduate school circa 1976)